郑州大学毕业设计（论文）

题目：基于指称表达式所指代物体的视觉问题回答系统设计

指导教师：田侦 职称：讲师

指导教师(校外)： 职称：

学生姓名：谢梓聪 学号：201824100445

专　　业：软件工程

院（系）：计算机与人工智能学院

完成时间：2022 年 5 月 12 日

2022 年 5 月 12 日

摘要

近年来深度学习的出现极大程度地促进了人工智能的发展，人工智能又极大地便利了人类的生活。在众多人工智能应用当中人机交互需求最为广泛，因此同时结合图像文本语音等多模态的人工智能变得越来越重要。文本是人类的高层抽象语义信息的载体，图像反映人类视觉系统的成像，如何同时结合文本和图像模拟和人类的实际交互是近年人工智能的一大难题。

视觉问题回答和指称表达式理解都是近年来视觉-语言人工智能研究领域的主流任务，视觉问题回答是指给定一副图像回答一个给定的与该图像相关的问题，指称表达式理解是指给定一副图像和一个指代某个图像物体的表达式来确定物体在图像的具体位置。

该篇论文旨在对于当前视觉问题回答任务的主流形式较为简单和较难用于实际的问题进行解决，提出一个新的双阶段任务形式，给定一副图片，先通过指称表达式指代一副图片中某个感兴趣的物体，若系统反馈找到了这样的物体，再提出一个对该物体的问题用于系统回答。该任务想法来源于真实的人机交互场景，可为盲人提供帮助，盲人可通过该任务的两次交互了解周围情况，同时该项任务可作为基础任务为后续高级人机交互任务如视觉语言导航提供很大帮助。针对提出的任务，本文使用了大型文本图像预训练模型 VL-BERT 在这两个下游任务的标准数据集上分别进行微调，再将两个任务的实时预测结合起来，并且做了一个双阶段的交互式的系统用于任务理解和结果可视化。综上可知本文研究成果以及贡献有如下三点：

（1）相比当前视觉问题回答形式提出更细致更易应用于实际人机交互场景的任务：基于指称表达式所指代物体的视觉问题回答。

（2）针对该任务提出使用大型文本图像预训练开源模型 VL-BERT 分别在指称表达式理解和视觉问题回答任务两个标准数据集上进行微调，在预测阶段将两个任务预测结果结合来完成该任务。

（3）针对提出任务和所用模型使用 Vue、ElementPlus 和 Django 开发一个交互式的前后端分离的 Web 系统用于任务理解和结果可视化。

关键词：视觉问题回答，指称表达式理解，人机交互，VLBERT，视觉与语言

Abstract

In recent years, the emergence of deep learning has greatly facilitated the development of artificial intelligence, which in turn has greatly facilitated human life. Among all of the AI applications, human-computer interaction is the most widely demanded, so it is becoming more and more important to combine multimodal AI at the same time. Text is the carrier of high semantic information reflecting human language, and image is the imaging reflecting human visual system, how to combine both text and image to achieve actual human interaction is a major challenge of AI in recent years.

Both visual question answering and referring expression comprehension are mainstream tasks in the field of visual-linguistic AI research in recent years. Visual question answering refers to answering a given question related to a given image, and referring expression comprehension refers to determining the specific location of an object given a image and an expression referring to an object in the image.

This paper aims to solve the problem that the current mainstream form of visual question answering tasks is simple and hard to use in practice. A new two-stage task is proposed, in which given an image, an object of interest is first referred by an referring expression, and if such an object is found by the system, then a question about the object is posed for the system to answer. This task is inspired from a real human-computer interaction scenario and can help blind people to understand their surroundings through the two interactions of this task. For the proposed task, we use a large text-image pre-training model, VL-BERT, to fine-tune two downstream tasks, referring expression comprehension and visual question answering benchmark datasets respectively, and then combines real-time predictions for both tasks, and makes a two-stage interactive heuristic system for task understanding and result visualization. In summary, the research results and contributions of this paper are as follows:

(1) A more detailed and easy-to-apply task for real human-computer interaction scenarios is proposed: referring expression comprehension based visual question answering.

(2) For this task we propose to use the large text-image pre-training open-source model VL-BERT to fine-tune it on referring expression comprehension and visual question answering benchmark datasets respectively, and then combine the predictions of the two tasks to accomplish the task.

(3) An interactive and heuristic front-end and back-end separated web system for task understanding and result visualization was developed using Vue, ElementPlus an

d Django for the proposed task and the model used.

Keywords: Visual Question Answering, Referring Expression Comprehension, Human-Computer Interaction, VLBERT, Vision and Language

目录

1 引言

1.1 研究背景

自从 2012 年卷积神经网络[1]出现后，深度学习得到了飞速发展，卷积神经网络、循环神经网络两大主流网络分别统治了图像和文本领域相关任务，如卷积神经网络广泛应用于图像分类、目标检测、语义分割和实例分割等任务，循环神经网络广泛应用于机器翻译、命名实体识别、情感分析、机器问答等任务，且深度学习的出现使得人工智能各个应用相比传统方法精度都得到了质的提升。但人工智能的需求并不仅限于单模态的文本或图像领域，如何解决多模态任务如图像和文本相结合的任务是实现真正意义上人机交互的不可或缺的基石。

近五年来视觉语言人工智能研究领域逐步发展，小到如视觉问题回答[2]、指称表达式理解[3]、图像描述等，大到视觉对话[4]、视觉语言导航[5]等，其中视觉问题回答是指给定一个图片对给定问题进行回答，指称表达式理解是指给定一个图片找到给定表达式指代的具体物体，模型使用从最简单的 CNN+LSTM 相结合到逐步嵌入多种注意力机制、模块化机制和图机制再到近期主流的大型文本图像预训练模型，这些任务的出现和成熟一定程度上促进了人机交互领域的发展，但是多数任务存在形式简单、与实际应用有差距等问题，在现有模型不断提升精度的情况下，需要有更加复杂以及贴合实际人机交互场景的任务提出，从而更准确地评估当前多个模型的精度以及落地难度。

1.2 研究内容

本文就是基于主流视觉语言任务中的视觉问题回答这一任务，针对其存在的形式简单单一、与实际人机交互场景差距较大的问题进行解决，提出一个双阶段的交互任务。给定一张图片，首先给定一个指定图片中某物体的表达式，由机器代理判断是否存在这样的物体，若存在这样的物体那再次给定一个与该物体相关的问题，并由机器代理回答，这实际上是将原来的指称表达式理解和视觉问题回答任务相结合的一个任务，通过指称表达式理解来达到更细致更实际的问题回答。将视觉问题回答常用标准数据集 VQA2.0[6]的任务与本文提出任务相比，前者问题类别主要分为计数类问题、判断图片场景的描述正确与否问题、颜色问题以及一部分常识推理类问题，后者则更注重于实际人机交互场景的问题，更能将问题聚焦于一个场景下多个物体中的一个具体物体。两者都着重于同时理解图像和文本，但后者相比前者更贴合实际交互场景，能够很好地在多个类似物体中聚焦到感兴趣的物体或区域，再通过一次问答更细致地了解该物体的情况。前者包含面广但真正意义上贴合实际人机交互场景的比例不

大，后者旨在解决能快速应用到实际的小部分问题而并未包含其他很多类问题。除了提出该项任务之外，本文还同时提出使用现阶段在指称表达式理解和视觉问题回答性能均表现极好的大型图像文本预训练模型，在这两个下游任务上进行微调后并在预测阶段将预测结果相结合来完成所提出任务。最后为了任务理解和结果可视化，本文同时提出开发一个前后端相分离的 Web 系统，使用 Vue 3 框架的 Composition API 和 ElementPlus 组件框架进行前端开发，使用 Vuex 做组件间的全局数据交互，使用 Vue Router 做组件的路径映射，使用 TypeScript 代替 JavaScript，使用 Axios 向后端发出异步请求，上述提到前端技术均是为了使得项目在开发过程中代码复用性强、可扩展性强以及可阅读性强。后端整体采用 Django 开发，具体使用 Django-Rest-Framework 做接口开发，使用 CORS-Headers 做跨域访问从而实现 Vue 前端与 Django 后端的交互。

1.3 研究意义

本文所研究工作的意义在于在现有视觉语言任务上提出更加贴合实际人机交互场景的任务，使得该任务上的研究成果能够快速在实际人机交互场景中应用。人工智能现阶段的前沿研究成果均存在落地难、与实际应用场景相差较大的问题，因此使得工业界人工智能的应用相比科研界人工智能的研究成果总要落后一阶段。本文在视觉问题回答这一任务上，使用指称表达式理解作为辅助任务来聚焦一个具体物体，从而实现在这个具体物体上的问题回答。该项任务可在后续应用于机器代理辅助盲人了解周围环境以及视觉语言导航等人机交互场景。

2 相关工作

2.1 指称表达式理解任务研究现状

指称表达式是指代某个场景下某一个具体物体的自然语言语句，表达式往往根据场景下某个物体的外观、属性、相对其他物体的位置和关系信息来确切指向一个物体，指称表达式理解是给定一副图片标出指称表达式所指代的具体物体边界框位置，是一个将计算机视觉和自然语言处理相结合的任务，需要同时理解文本和图像来进行预测，具体任务样例可参阅图 2.1。不同于目标检测提前定义好所有目标的类别，指称表达式理解所要指代的物体类别在预测时才能根据表达式确定，因此指称表达式理解比传统计算机视觉问题更加困难。指称表达式理解这一任务从提出以来逐步发展出很多方法与模型[7]，主要分为使用 CNN+LSTM 将图片和文本特征嵌入到一个特征空间的传统方法[3], [8], [9]、在传统方法上添加模块化机制[10], [11]、图机制[12]、注意力机制[10], [11], [13]的改进方法、使用额外解析器的模型[14]和大型图像文本预训练模型[15], [16]等。该任务常用于训练和评估的基准数据集有 RefCOCO、RefCOCO+以及 RefCOCOg 等，其中前两者是通过一个双人交互式游戏在 MSCOCO 通用目标检测大型自然图像数据集上进行收集的，即玩家一基于图片特定物体写出指称表达式，仅给定玩家二图片和表达式让其点击物体区域，若在真实物体区域内则两者均得分且交换身份，若不正确在仍重复上述过程，最终留下两人均得分的表达式并记录对应图片。前两者主要区别是 RefCOCO+限制了不能使用物体在图片中的绝对位置，这使得 RefCOCO+更趋向于使用外观特征和物体属性值去指向某物体，而 RefCOCOg 相比前两者是使用非交互方式获取的，整体表达式的长度和复杂度均有提高。具体各类方法在数据集 RefCOCO+[8]测试集上表现如表 2.1 所示，其中表现最好的模型 ViLBERT[15]在该任务上的精度已经足够应用于实际当中。

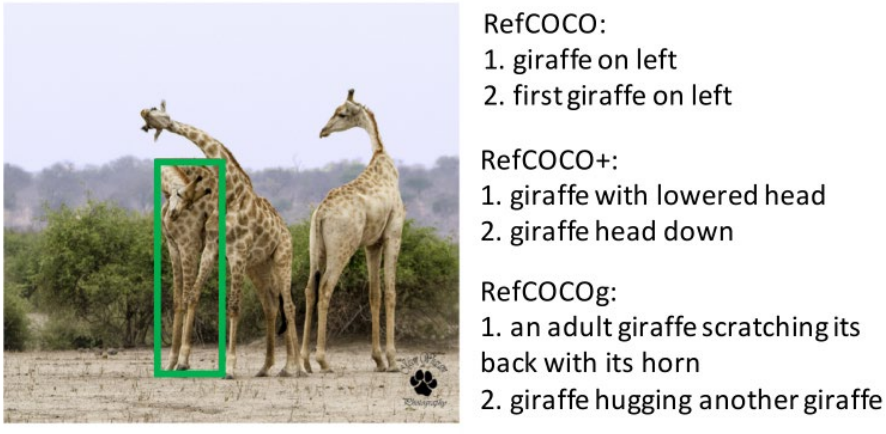

图 2.1 指称表达式理解任务举例

表 2.1 使用检测候选物体时上述模型在 RefCOCO+的 IOU 比较

模型	图像特征提取模型	分类	testA	testB
MMI[3]	vgg16	CNN/LSTM	54.03	42.81
Visdif[8]	vgg16	CNN/LSTM	50.10	37.48
Spe+Lis+RI[9]	frcnn-resnet101	CNN/LSTM	58.68	48.23
CMN[10]	vgg16	模块化和注意力机制改进	54.32	47.76
MattNet[11]	mrcnn-resnet101	模块化和注意力机制改进	71.62	56.02
LGRANs[12]	frcnn-vgg16	图机制改进	64.0	53.4
CM-Att-Erase[13]	frcnn-resnet101	注意力机制改进	73.65	58.03
NMTree[14]	frcnn-vgg16	额外解析器改进	72.02	57.52
ViLBERT[15]	frcnn-resnet101	图像文本大型预训练	78.52	62.61
VLBERT[16]	frcnn-resnet101	图像文本大型预训练	77.72	60.99

2.2 视觉问题回答任务研究现状

视觉问题回答是计算机视觉和自然语言处理交叉领域的一个任务，是给定一副图片和一个与该图相关的问题，对该问题进行回答的任务，它需要结合图片中的视觉元素和一些常识甚至是知识库来正确回答问题。问题类型主要分为正确与否的二分类问题、多选一选择答案的多分类问题、完形填空式问答的多分类问题以及开放性的生成问题，相比前三者是分类问题即答案集合已经预先定义好，后者是开放性的问题，需要使用符合句法结构的自然语言短语来回答，标准数据集上常将该类问题分为物体类别判定、场景某物体数量计数、物体颜色判定以及物体位置判定等，具体样例可参阅图 2.2 和图 2.3。总体来看，现阶段视觉问题回答任务中问题语义信息量较少形式较为单一，若应用于实际的话会存在应用场景较少的问题。因此在传统视觉问题回答之上提出了很多更复杂更有应用场景的视觉问答任务如视频问题回答、交互场景下的视觉问答[17]、

基于事实[18]和知识库[19]的视觉问答、视觉对话[4]等等。视觉问题回答这一任务自 2015 年提出以来也发展出了各类模型用于解决不同形式的上述提到的视觉问题回答任务[20]，主要分为使用 CNN+LSTM 将图片和文本特征嵌入到一个特征空间的传统方法、在传统方法上基于注意力机制、可分解式模型、额外知识库的改进方法以及大型文本图像预训练模型等，具体各方法在该任务常用标准数据集 VQA2.0[6]测试集上表现如表 2.2 所示，虽然在该数据集上表现精度较高，但由于现有数据集均存在数据偏差、语言先验以及问题形式不够丰富、语义结构不够丰富等问题，仍不足够应用到实际人机交互场景当中。

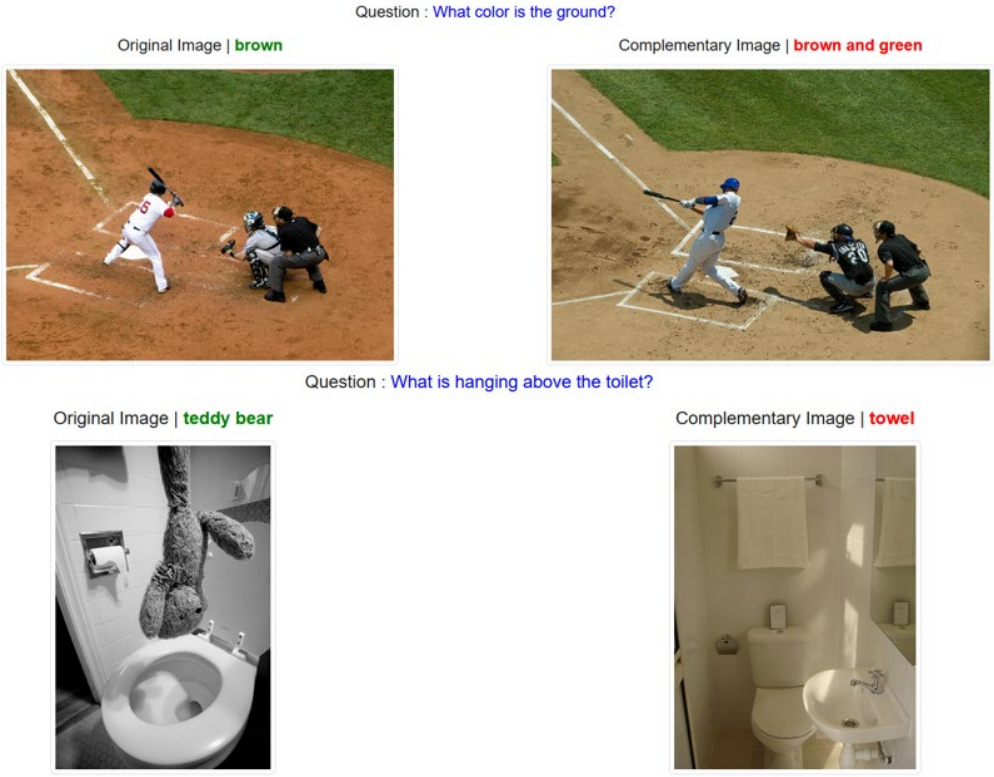

图 2.2 视觉问题回答任务 VQA2.0 举例（上：颜色，下：物体）

表 2.2 各视觉问题回答模型在 VQA2.0 上的精度比较

模型	分类	test-dev	test-std
BUTD[26]	注意力机制改进	65.32	65.67
ViLBERT[16]	大型预训练	70.55	70.92
VisualBERT[27]	大型预训练	70.80	71.00
LXMERT[28]	大型预训练	72.42	72.54
VL-BERT[15]	大型预训练	71.19	72.22

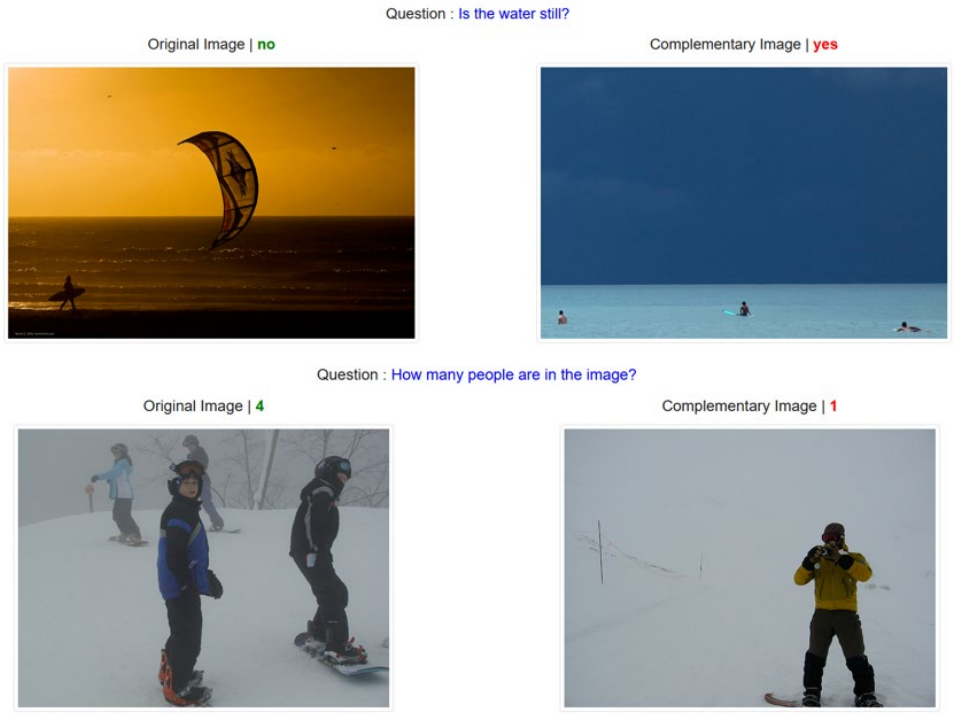

图 2.3 视觉问题回答任务 VQA2.0 举例（上：是否，下：数量）

2.3 其他视觉语言任务研究现状

随着计算机视觉和自然语言处理在近年来得到了很大发展，以及多模态数据在自然界普遍存在，因此视觉语言相结合多模态方向的人工智能研究在近年来发展迅猛且仍在持续增长，除了本文提到的视觉语言问答和指称表达理解之外，还有其他很多正在持续研究的视觉语言任务，但是视觉语言各项任务都会面临图像和文本两个模态间存在语义鸿沟的挑战[20]：图像是高维的且比纯文本含有更多噪声；图像缺乏语言所含有的句法结构和语法规则且没有自然语言中可直接做句法解析和正则表达式匹配的工具包；图像捕捉真实世界的丰富性而自然语言已经达到高水平的抽象。其他相关视觉语言任务诸如视觉常识推理[21]、图像描述、视觉语言导航[5]等任务，其中视觉常识推理是指给定一个有关某个图像的问题，机器必须正确回答且提供一个合理的理由证明自己的答案；图像描述是指给定一个图片给出有关该图片内容的自然语言描述；视觉语言导航是指给定一条导航指令让机器代理在 3D 室内全景场景下进行导航。这些任务在提出以来都获得了很大的提升，但是与视觉问题回答一样，视觉语言任务的关键是要解决图片和文本的特征融合，而图片和文本之间的语义鸿沟一定程度上限制了各类视觉语言模型的性能，因此这些任务往往存在形式上的限制，如视觉常识推理局限为四选一的选择问题，以及视觉语言导航在室内场景离散化后的图上面导航，这些局限表明了当前大多数视觉语言模型还无法像人类一样完成高难度的视觉语言理解，因此视觉语言还处在一个需要持续得到发展的阶段。

2.4 图像文本大型预训练及微调研究现状

自 Transformer[22]提出完全使用自注意力机制代替 RNN 来完成自然语言处理任务以来，使用 Transformer 做文本大型预训练便大放光彩[23]，在通用型数据集上做大型自监督预训练使得模型学得某个领域的泛化特征，并在若干个该领域下游任务标准数据集上做微调便可在各个下游具体任务上达到极佳的性能。这种使用 Transformer 做大型预训练很快也被应用到图像上[24]，并代替传统 CNN 架构来完成泛化特征提取并迅速应用到各下游任务上。随着计算机视觉和自然语言处理领域 Transformer 的大放异彩，这种大型预训练的方法很快被应用到视觉语言领域[15], [16]，通过在大量图像文本对上做大型预训练，然后在若干个下游任务上如视觉问题回答、指称表达理解等标准数据集上做微调即可达到很好的效果，本文使用 VLBERT[16]作为模型即是利用了大型预训练的优点，使用一个模型微调又同时在两个任务上都达到很高的性能，从而更快速的完成所提出的任务。

3 系统设计

3.1 总体设计

系统的整体流程是：用户通过在前端表单从图片库中浏览选取或上传图片并输入指称表达式后提交一阶段的指称表达理解任务请求，后端部署好模型和接口的 GPU 服务器响应该次任务请求，并在后端执行一次模型前向输入，得到结果后返回给前端，二阶段的视觉问题回答任务请求执行过程与一阶段过程一致。

这个过程与以往普通接口不同的是，前端向后端发起了一次任务请求，后端接到请求后不仅需要操作数据库，还需要执行一次模型预测过程，而对于 VLBERT 大型预训练模型而言每加载一次就需要花费二十秒以上的时间，假如使用最简单的方法在后端操作数据库之前先加载模型后预测最后写入数据库的话，每次请求则需要二十秒以上的响应时间，这无论是对于本文用于演示的系统还是将来应用到实际的系统而言均不合理。

因此，本文考虑使用 RabbitMQ 去构建一个任务请求消息队列，将指称表达理解任务和视觉问题回答任务分别作为一个生产者去对待，将每次前端发来的请求作为消费者对待，生产者预先加载好模型，后台运行直至对应的任务消息队列不为空，每次读取一次任务然后模型前向预测一次，并在任务处理结束得到结果后通过 WebSocket 通信传达结果反馈给前端，而前端页面初次加载时就连接后端 WebSocket 服务器确保连接，当 WebSocket 客户端实例消息回调函数触发时，将结果渲染给前端组件，以完整实现上述流程。这里使用 WebSocket 的原因在于模型预测消耗时间不定，且消费者执行任务是异步的，因此需要服务器在执行完任务后主动向前端发送一次消息。前端使用 Vue3+ElementPlus 进行组件式的模块化界面设计，后端使用 Django+Django Rest Framework 进行快捷低代码的接口设计。整体系统框架图如图 3.1 所示，数据流图如图 3.2 所示。

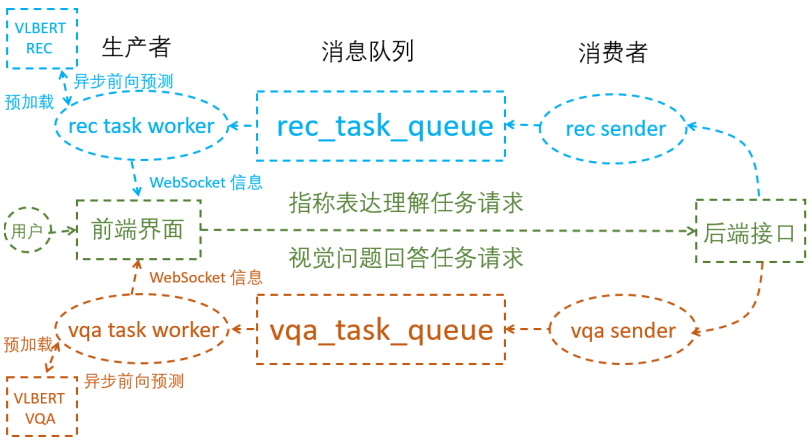

图 3.1 系统框架图

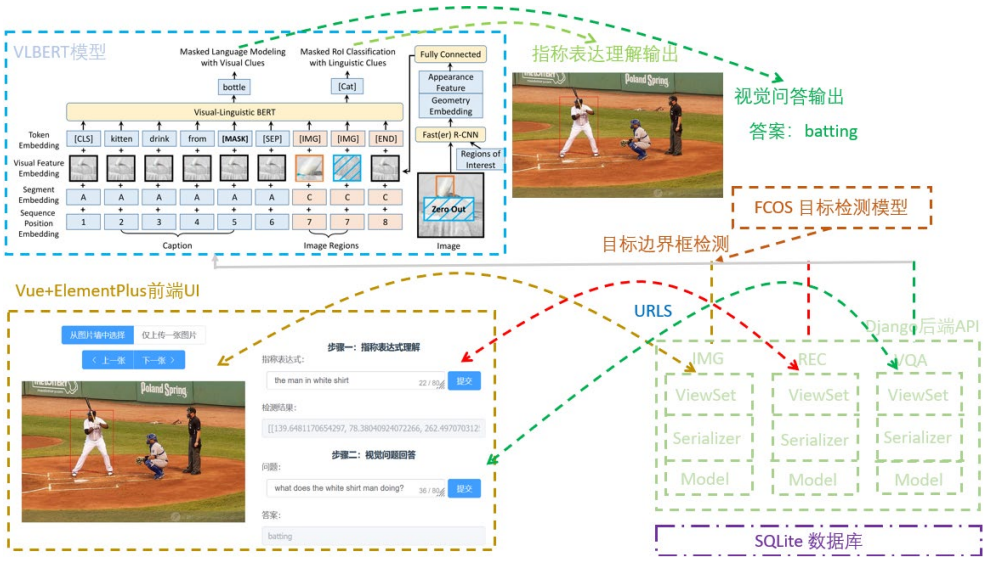

图 3.2 数据流图

3.2 任务设计

因本文上述所提到的当前视觉问题回答存在问题形式局限、问题语义结构不够丰富、问题不够精确等问题，现有模型较难快速应用到有实际需求的人机交互场景中，故本文提出一个新型双阶段的交互式问答任务：基于指称表达式所指代物体的视觉问题回答，即先通过指称表达式理解这一辅助任务聚焦在感兴趣的物体上，再通过具体的视觉问答进一步了解该物体的情况。具体是先通过明显的外观和属性描述、相对其他明显物体的位置和关系描述来明确指代某个物体，后通过询问该物体详细外观信息来了解该问题更多信息，具体一个完整的双阶段任务可参照图 3.3。

该任务启发于实际人机交互场景，想象一个盲人在一个室内或室外场景下，他需要机器代理先使用自然语言描述一下所看到的场景，这其中会包含一些物体的外观属性和一些物体间的相对位置和关系信息，因此这里选择使用明显的外观属性和相对位置和关系信息来指代，因为现有图像描述模型结果往往还不足够精确到很详细的属性，然后盲人为了了解周围情况需要对各个场景物体做更详细的了解，此时便需要先指代感兴趣的物体再详细地问该物体的外观信息。又比如在智能家居应用中，服务机器人要辅助行动不便老人看一些场景的实时情况，则老人可根据自己对场景的熟悉指代机器到某一场景下查看并询问该物体的详细情况。

该任务相比于当前视觉问题回答常用基准数据集[6], [25]的任务，之所以更贴合实际应用场景，是因为它能在一个场景多个不同类物体或同类不同外观物体之间进行区分后指代感兴趣的物体进行视觉问答，它能够以不含歧义的方式精确指代感兴趣的物体，而不是仅仅宽泛的问整个场景的一些正确与否问题来

了解整个场景，启发于人类生活中的场景问答习惯，采用多次做一个双阶段先指代某物体再具体询问的任务来使其更贴合实际人机交互场景。

图 3.3 基于指称表达式所指代物体的视觉问题回答任务举例

3.3 算法设计

3.3.1 预训练模型微调

为解决上述所提出任务，需要同时可以完成指称表达式理解和视觉问题回答的模型用来做实时预测，在前期调研视觉问题回答[20]、指称表达理解综述[7]后，选择使用现阶段表现性能极佳且能同时应用到这两个下游任务的大型文本图像预训练模型 VL-BERT 模型[16]在这两个下游任务基准数据集上进行微调。

考虑到 VLBERT 原模型是使用 8 卡 16G V100 进行分布式多卡预训练和微调的，且安培架构的 GPU（30 系列、A40、A4000 等）需使用 CUDA11.x 及以上版本的 CUDA，而 VLBERT 需要使用 CUDA9.0 做一些模块编译，因此需要选用 Tesla 系列能使用 CUDA9.0 的 GPU，最后租用了 AutoDL 平台的 32G V00 GPU1 卡 17 小时、7 卡 17 小时，分别在指称表达理解常用基准数据集 RefCOCO+[8]和视觉问题回答常用基准数据集 VQA2.0[6]上做轮次为 4 的训练，微调时的训练超参数选择参照 VLBERT，同时在 32G GPU 上，可将 batch size 做适应性的修改，分别将两个下游任务 batch size 扩大两倍，视觉问题回答微调阶段 batch size 使用 8，采用 Adam 优化器，学习率为1×10^{-4}，$\beta_1 = 0.9$，$\beta_2 = 0.999$，权重衰减为10^{-4}，学习率在最开始 2000 步内采用学习率线性衰减进行学习率预热，使用在 Visual Genome 预训练的 Faster RCNN 在 MSCOCO 上的检测框。指称表达理解微调阶段 batch size 使用 8，累计梯度步数为 2，即总 batch size 为 16，学习率为1×10^{-4}，$\beta_1 = 0.9$，$\beta_2 = 0.999$，权重衰减为10^{-4}，学习率在最开始梯度下降 500 步内采用学习率线性衰减进行学习率预热，使用 MAttNet 预训练的 MaskRCNN 的 Fast RCNN 分支在 MSCOCO 上的检测框。具体微调结果与原论文对比请参照表 3.1。

表 3.1 VLBERT 在指称表达理解和视觉问答上微调结果与[16]精度对比

任务	batch size	微调时间	优化器	学习率	β_1	β_2	VLBERT 精度	本文微调精度
VQA	56	7 卡 17h	Adam	10^{-4}	0.9	0.999	71.16	70.93
REC	16	1 卡 17h	Adam	10^{-4}	0.9	0.999	77.72/60.99	76.44/60.63

3.3.2 双阶段任务预测

VLBERT 源代码中分别设计了指称表达理解和视觉问题回答的 train、val 和 test 的模型输入输出代码，将 RefCOCO+表达式数据、MSCOCO 图片以及标注数据、FastRCNN-ResNet101 在 MSCOCO 图片上预先计算好的检测框数据封装为 RefCOCODataset，将 VQA2.0 问题以及答案标注数据、MSCOCO 图片以及标注数据、在 Visual Genome VQA 数据集预训练好的 FastRCNN-ResNet101 特征、在 MSCOCO 上预先计算好的检测框以及图像特征数据封装为 VQADataset，然后通过这两个 Dataset 构建 Pytorch 的 DataLoader 在 train、val、test 模型前向输入时直接获取定义好的数据，与源代码不同的是，预测单个样本的代码仅需要前端请求发来的单个图片以及文本，因此不需要定义 Pytorch 的 Dataset，但为了将数据能输入到 VLBERT 中，需要将文本经过预训练 BERT 进行 Tokenize 并转化为 ID，并通过 Pillow 图像处理库读取图片到内存中。

除此之外，VLBERT 在上述两个下游任务上预测时还需要预先通过 Faster-RCNN 等目标检测网络检测出图片的所有候选物体即可能感兴趣的物体，然后将检测出所有物体的边界框、图片和文本输入到微调后的模型中做预测。

考虑到 FasterRCNN 作为双阶段目标检测网络以及现有开源仓库所需 Python 环境以及依赖库版本均与 VLBERT 不兼容，本文采用 FCOS 单阶段目标检测网络加载预训练好的模型完成候选框的检测，VLBERT 在 Python3.6、Pytorch1.1、TorchVision0.3 和 Cuda9.0 虚拟环境中运行，在此环境上通过 FCOS GitHub 仓库进行 pip 安装到环境内部，指称表达式理解和视觉问题回答单次请求各个阶段平均耗时如下表 3.2 所示。

表 3.2 Tesla P40 上指称表达理解和视觉问答预测各阶段平均耗时

任务/平均耗时	FCOS 目标检测	加载图片	BERT 文本处理	VLBERT 前向	总耗时
指称表达理解	25ms	2ms	5ms	200ms	232ms
视觉问答	25ms	2ms	5ms	220ms	252ms

3.4 网页设计

3.4.1 前端设计

1. 组件设计

前端设计使用 Vue.js 框架进行开发，Vue.js 框架是一种面向前端界面开发的轻量化框架，其设计模式遵循自下而上的原则，此种开发框架的最本质的特点是具有响应式编程和组态化的特点[29]。本系统基于 Vue3 开发了两类组件，分别是内容填充组件和布局组件，其中内容填充组件包含图像填充组件、上传文件组件、切换图片组件、指称表达理解任务表单提交组件、视觉问题回答任务表单提交组件、照片墙组件、提交历史查询组件，布局组件包含顶部标题栏、左侧标签页栏、中部内容栏和底部版权栏，具体各个布局组件如何布局和内容组件如何填充可见图 3.4 所示。使用 Vue 所提供的组件思想去开发整个网页的模块化布局和内容组件有利于提升代码可复用性、可阅读性和可扩展性。

使用 Vuex 作为各个组件间的数据共享读写库来实现任意组件间的数据收发。使用 Vue Router 实现路径到哪些内容组件填充至中部布局组件的映射，此插件是为 Vue.js 框架提供路由管理的插件，借助 hash 和 History interface 两种方式实现前端路由[30]，具体映射情况如表 3.3 所示。

表 3.3 Vue Router 中组件和路径之间的映射关系

包含组件	URL
ImageHolder、SwitchHolder/UploadHolder、RecForm、VqaForm	/app/Main/
RequestHistory	/app/Request/
PhotoWall	/app/PhotoWall/
SettingPanel	/app/Setting/

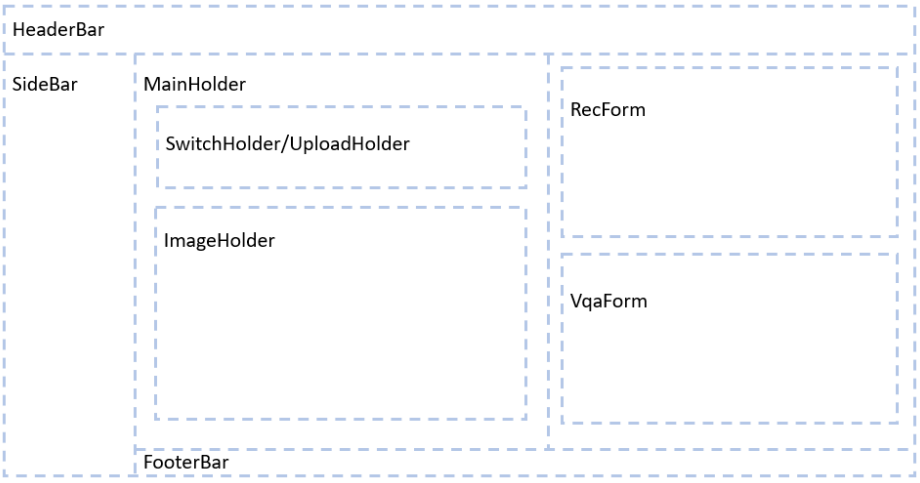

图 3.4 组件间布局和填充关系

各个路由中 Main 路由将图片显示组件、切换图片或上传文件组件、指称表达理解和视觉问题回答表达提交组件填充至 MainHolder 主体内容布局组件中，Request 路由将请求历史组件填充至 MainHolder 中，Request 路由下请求具体某个任务处理细节时将请求详情组件填充至 MainHolder 中，PhotoWall 路由将照片墙组件填充至 MainHolder 中，Setting 路由将设置表单填充至 MainHolder 中。使用 Vue 开发独立组件的方式，实现了顶部标题栏、侧边标签栏和底部版权栏在路由改变时不会再次得到渲染，而只根据对应路由重新渲染改变组件的内容。通过 SideBar 布局组件的标签栏实时设置 Vuex 全局 store 中的 Mode，并通过 Vue Router 和内容填充布局组件的数据绑定 Mode 来实现侧栏更换标签页并更新 URL。

2. 主操作界面设计

主操作界面适用于上传图片或者从照片库中左右浏览并选取图片后先填写指称表达理解表达并发出请求，后端响应反馈存在所指代物体时，左侧图片填充栏会将检测出的所指代物体框选出来，并将检测结果置于指称表达理解表单结果一项中，再填写视觉问题回答表达表达并提交，等待后端响应反馈问题答案，响应后将答案填充在视觉问题回答表单中。可通过单选按钮选择是浏览选择已上传到图片库中的图片还是当前通过文件上传器上传一张新的图片，上传时也会更新到图片库中，图片上传是直接使用上传时文件进行命名，因此不可重复上传相同文件名称的图片。具体操作界面如图 3.5 所示。

图 3.5 主操作界面设计

3. 图片墙界面设计

图片墙的存在是为了不持续上传同副想要使用的照片，无论在主界面上传还是照片墙上传均可上传到图片库中并存储在后端服务器中。具体实现是基于 Element Plus 的图片 Grid 组件，可进行单个图片预览、上传和删除。具体界面如

图 3.6 所示。

图 3.6 图片墙界面设计

4. 请求历史以及详情界面设计

请求历史是为了查看之前所有请求的预测结果，无论是单次仅请求了指称表达理解的请求项，还是接连请求了指称表达理解和视觉问题回答的完整请求项，都会展示在该界面的数据表格当中，当针对一个指称表达请求进行了多个视觉问题回答请求时，也可以通过下拉栏查看某个指称表达请求项的所有后续视觉问题项。无论是单次请求还是接连双阶段请求都可再次点击对应的详情链接跳转到具体的历史操作界面，用于展示请求图片、请求的指称表达式、问题以及预测结果。若仅进行了单次请求，还可在详情主操作界面进行后续视觉问题回答的请求。具体界面如图 3.7-3.9 所示。

图 3.7 请求历史查询数据表格界面设计

图 3.8 指称表达理解单次请求详情界面设计

图 3.9 视觉问题回答接连双阶段请求详情界面设计

5. 设置界面设计

设置界面从简设计，因上述提到任务和模型当中多数术语为英文，因此为保证用语一致性在整体前端界面均采用英文显示，可通过设置入口设置中文界面。

6. 设计模式

整体采用 TypeScript 代替 JavaScript 进行开发，使用 SFC（Single File Component）和 Composition API 进行开发，使用 Axios 代替 Ajax 向后端发出 Promise 形式的 Restful 请求，一定程度上提高了代码简洁性和可阅读性。前端组件库方面使用基于 Vue3 开发的 Element Plus 组件库，一定程度上美化了界面设计，且由于 Element Plus 也是采用模块化的设计思想设计的各个常用业务组件，因此同时提高了代码可复用性和可扩展性。总体来看，前端使用 Vue3 结合 Element Plus 的方式去进行开发，并使用很多开发模式、开发框架和开发包去提高前端代码的可复用性、可扩展性、可阅读性、鲁棒性和简洁性。

3.4.2 后端设计

1. 设计模式

由于上述模型部署时需要使用 Python 环境，因此考虑使用 Python 方面 Web 开发比较成熟的框架 Django 来做接口开发，Django 是一个开放源代码的 Web 应用框架，由 Python 编写，采用了 MTV 的框架模式，即模型 M，视图 V 和模版 T[31]。这里仅使用了 M 和 V 部分做接口开发，V 由前端 Vue 应用实现，同时结合使用 Django-Rest-Framework 去做 Django 代码的简化并可快速开发出对各个数据库表的增删改查的可浏览式接口，同时提高代码的简洁性、可复用性和可扩展性。

2. 数据库设计

数据库方面由于设计数据库实体表较少，因此直接使用 Python 内置的 SQLite 数据库，具体实体表包含用于图像上传的 IMG 表、用于提交指称表达理解任务请求的 REC 表、用于进一步提交视觉问题回答任务请求的 VQA 表，其中 VQA 和 REC 间存在一对多的联系，即每个 VQA 实体需要依托于一个 REC 实体存在，一个 REC 实体可以有多个对应的 VQA 实体。数据库物理模式设计如下表 3.4-3.6 所示。

表 3.4 IMG 表物理模式设计

字段	数据类型	注释
id	integer	主键
img	varchar(100)	非空

表 3.5 REC 表物理模式设计

字段	数据类型	注释
id	integer	主键
socket_id	varchar(255)	非空
referring_expression	varchar(255)	非空
result	varchar(255)	允许空
result_image	varchar(255)	允许空
img_id	integer	外键至 IMG

表 3.6 VQA 表物理模式设计

字段	数据类型	注释
id	integer	主键

socket_id	varchar(255)	非空
question	varchar(255)	非空
answer	varchar(255)	允许空
rec_id	integer	外键至 REC

3. 接口设计

后端接口开发主要基于 Django-Rest-Framework 提供的三层架构来写，即分 models、serializers、viewsets 三层来写，models.py 包含数据库实体表的各字段定义，serializers.py 包含数据库增删改查的序列化操作，viewsets.py 包含 Serializer 序列化后的数据以及需要渲染到的 HTML 模板，这里主要需要对 Serializer 得到的序列化数据做一些过滤处理等操作而不需要模板渲染，即在增删改时复写 Serializer 和 Viewset 的一部分函数，最后在 urls 中去做 viewsets 和访问 url 之间的映射即可。Models 内部定义即如上述数据库物理模式一致，Viewset 实现当中 IMG 上传时为了满足不上传重复名称的图片则需复写 create 函数，查询是否已有该图片，若没有再添加并上传该图片，当删除 IMG 时同时删除 MEDIA 路径下之前上传的图片，具体 models、serializers 和 viewsets、urls 代码如下图 3.10-3.13 所示。

使用 Django-Rest-Framework 的好处除了使用极少数分层式代码就可开发出对应实体表的增删改查之外，还提供了用于快速测试 API 的浏览式 API 界面，具体可参看图 3.14-3.16。

```
class IMG(models.Model):
    img = models.ImageField(upload_to='custom/')
    def __str__(self):
        return self.img.name
    class Meta:
        db_table = 'img'
        ordering = ['id']
        verbose_name = 'IMGTable'
        verbose_name_plural = verbose_name

class REC(models.Model):
    referring_expression = models.CharField(max_length=255, verbose_name='referring_expression')
    img = models.ForeignKey(IMG, on_delete=models.CASCADE)
    result = models.CharField(max_length=255, verbose_name='result', null=True, blank=True)
    result_image = models.CharField(max_length=255, verbose_name='result_image', null=True, blank=True)
    def __str__(self):
        return self.referring_expression
    class Meta:
        db_table = 'rec'
        ordering = ['id']
        verbose_name = 'RECTable'
        verbose_name_plural = verbose_name

class VQA(models.Model):
    question = models.CharField(max_length=255, verbose_name='question')
    answer = models.CharField(max_length=255, verbose_name='answer', null=True, blank=True)
    rec = models.ForeignKey(REC, related_name='vqas', on_delete=models.CASCADE)
    def __str__(self):
        return '%s: %s'.format(self.rec.referring_expression, self.question)
    class Meta:
        db_table = 'vqa'
        ordering = ['id']
        verbose_name = 'VQATable'
```

图 3.10 models 设计代码

```python
class VQASerializer(serializers.ModelSerializer):
    class Meta:
        model = VQA
        fields = ('id','socket_id','question','answer','rec')

class RECSerializer(serializers.ModelSerializer):
    vqas = VQASerializer(many=True, read_only=True)
    class Meta:
        model = REC
        fields = ('id','socket_id','referring_expression','img','result','result_image','vqas')

class IMGSerializer(serializers.ModelSerializer):
    class Meta:
        model = IMG
        fields = ('id','img')
```

图 3.11 serializers 设计代码

```python
class RECViewSet(viewsets.ModelViewSet):
    queryset = REC.objects.all()
    serializer_class = RECSerializer
    def perform_create(self, serializer):
        validated_data = serializer.validated_data
        print('rec validated_data', validated_data)
        try:
            img_name = validated_data['img'].img
            referring_expression = validated_data['referring_expression']
            socket_id = validated_data['socket_id']
            image_path = MEDIA_ROOT + '/' + str(img_name)
            log_to_terminal(socket_id, {'terminal': 'Starting REC job...'})
            sender_rec(str(image_path), str(referring_expression), str(socket_id))
        except:
            log_to_terminal(socket_id, {'terminal': traceback.print_exc()})
    

class VQAViewSet(viewsets.ModelViewSet):
    queryset = VQA.objects.all()
    serializer_class = VQASerializer
    def perform_create(self, serializer):
        validated_data = serializer.validated_data
        print('vqa validated_data', validated_data)
        try:
            question = validated_data['question']
            rec = validated_data['rec']
            socket_id = validated_data['socket_id']
            log_to_terminal(socket_id, {'terminal': 'Starting VQA job...'})
            sender_vqa(str(question), str(rec), str(socket_id))
        except:
            log_to_terminal(socket_id, {'terminal': traceback.print_exc()})

class IMGViewSet(viewsets.ModelViewSet):
    queryset = IMG.objects.all()
    serializer_class = IMGSerializer
    def perform_destroy(self, instance):
        filename = settings.MEDIA_ROOT / instance.img.name
        print(type(filename), filename)
        if filename.exists():
            filename.unlink()
        instance.delete()
    def perform_create(self, serializer):
        validated_data = serializer.validated_data
        print('validated_data', validated_data)
        filename = 'custom/' + validated_data['img'].name
        print(filename)
        try:
            instance = IMG.objects.get(img=filename)
        except:
            return serializer.save()
```

图 3.12 viewsets 设计代码

```
router = DefaultRouter()
router.register(r'recs', views.RECViewSet)
router.register(r'vqas', views.VQAViewSet)
router.register(r'imgs', views.IMGViewSet)
urlpatterns = [
    url(r'^', include(router.urls))
]
```

图 3.13 urls 设计代码

图 3.14 浏览式 API 查看指称表达请求的列表界面

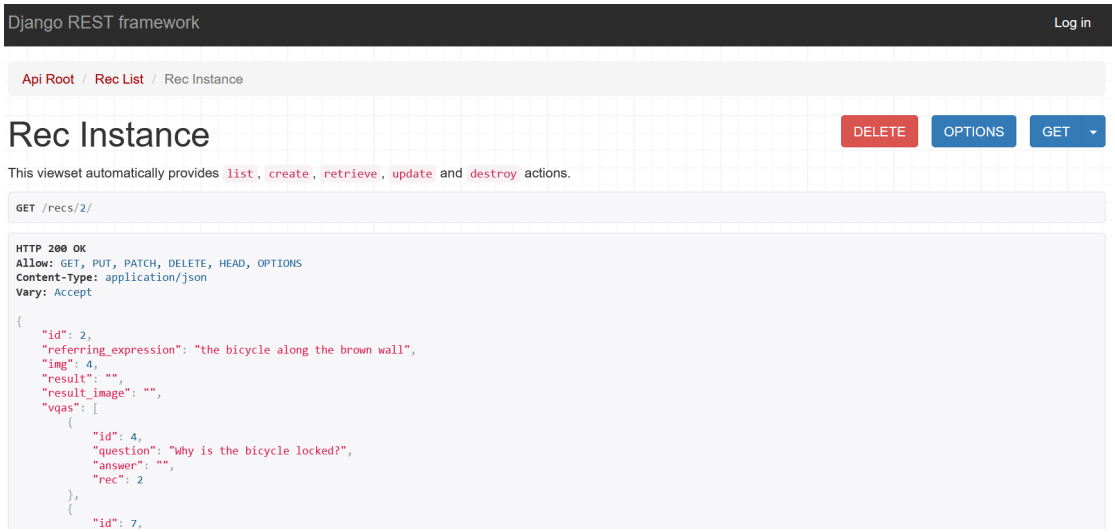

图 3.15 浏览式 API 改查指称表达请求实例界面

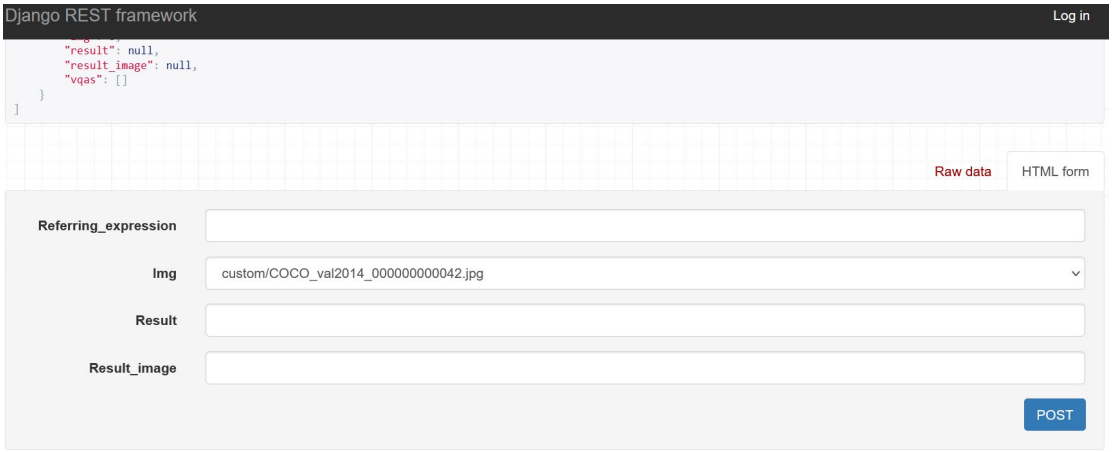

图 3.16 浏览式 API 增加指称表达请求界面

同时为了使 Vue 开发的 8080 前端应用和 Django 开发的 8000 后端应用能够进行数据交互即前端向后端发送请求后端向前端相应的过程，还需要使用 CORS-Headers 实现跨域访问，从而完成整个系统的实现。

3.5 开发与部署环境

3.5.1 开发环境

开发环境前端和后端均是基于 VSCode 进行开发，前端基于 Node.js 构建 JavaScript 环境、基于 npm 淘宝源做包管理工具，需安装 vue3、elementplus、vuex、vuerouter、axios、uuid 和 typescript；后端基于 Conda 建立虚拟 Python3.6 环境，使用 pip 清华源安装 django、django-rest-framework、cors-headers 以及 VLBERT 模型依赖包。

3.5.2 部署环境

部署后端由于需要租用 AutoDL 平台 Tesla 系列 P40 GPU 服务器，但租用平台的服务器实例只允许开放固定一个端口 6006，因此不能在该 GPU 服务器上同时占用前端 Vue 应用的 8080 和后端 Django 应用的 8000，故考虑使用 AutoDL 平台 GPU 服务器部署后端 Django 应用，使用阿里云免费服务器部署前端 Vue 应用。后端服务器系统为 Ubuntu18.04，其他环境为 Cuda9.0，cuDNN7.6.5，gcc 4.9.3。前端服务器环境为 Ubuntu20.04，其他环境为 Node.js16.15.0，npm8.5.5。

4 总结与展望

4.1 总结

第一，本文相比传统视觉问题回答任务提出了一个新的能更快速应用于实际应用场景的双阶段视觉问题回答任务，首先通过指称表达式指代图片中的一个具体物体，若存在这样的物体则再提出与该物体相关的问题让系统回答。该任务能在一个场景下众多物体中首先允许系统聚焦在感兴趣的物体上，然后再通过问答了解这个物体更详细的信息，符合实际人机交互的场景，能更快速用于实际中。

第二，本文通过 VLBERT 大型图像文本预训练模型分别在指称表达理解和视觉问题回答这两个下游任务上进行微调，并在实时预测时先通过指称表达理解找到物体，得到边界框后将该物体视觉特征、图片视觉特征和问题输入到视觉问题回答模型用于最后输出，使用相对其他物体位置关系以及联系或简单的外观属性来定位目标，再通过问答详细了解具体的其他外观属性。

第三，通过 Vue 和 Django 开发前后端分离的 Web 系统用于任务理解和结果可视化，在前后端均使用相关技术来提高代码可复用性、可扩展性、鲁棒性和可阅读性，并通过前后端分离方式来部署系统。

4.2 展望

提出上述任务后，使用上述方法的准确率较低，一部分误差来源于一阶段指称表达理解边界框的偏差，另一部分来源于二阶段视觉问答的答案误差，同时由于使用大型预训练模型，会导致预测耗时长要求 GPU 性能强。对于问题一，考虑使用更强的视觉特征提取模型来提取图像和候选框的特征。对于问题二，考虑针对该任务在已有数据集标注上做一些适应性修改形成新提出任务上的数据集并在该数据集上微调而不是在 VQA2.0 上微调，并提出对应的评价指标用于更好的模型收敛。对于问题三，一方面考虑在中低性能 GPU 设备上使用模型压缩和模型蒸馏来降低模型所耗显存和提高模型推理速度，另一方面考虑降低图片分辨率和文本嵌入向量的长度从而降低预测阶段数据并行所消耗的显存。

除此之外，该任务并未包含所有类型的视觉问答，只是提出一类可以快速用于实际人工智能场景的任务，且该任务只是作为整个场景交互的一部分，若要真正完成人机交互，还需要和其他很多视觉语言任务相结合，如在辅助盲人视觉语言导航过程中，可能需要先做场景描述，描述后盲人通过指称表达指代某具体场景物体，便可后续详细了解该物体，导航至该物体或对该物体执行某个操作。总体而言，该任务在一定程度上对逐步实现真正意义上人机交互具有

启发意义，通过后续在该任务上做改进工作，对实现自主型人工智能机器代理具有现实意义。

参考文献

[1]A. Krizhevsky, I. Sutskever, and G. E. Hinton, 'ImageNet Classification with Deep Convolutional Neural Networks', p. 9, 2012.

[2]A. Agrawal et al., 'VQA: Visual Question Answering', arXiv, arXiv:1505.00468, Oct. 2016. doi: 10.48550/arXiv.1505.00468.

[3]J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, 'Generation and comprehension of unambiguous object descriptions', in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 11–20.

[4]A. Das et al., 'Visual dialog', in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 326–335.

[5]P. Anderson et al., 'Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments', in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, Jun. 2018, pp. 3674–3683. doi: 10.1109/CVPR.2018.00387.

[6]Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, 'Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering', arXiv, arXiv:1612.00837, May 2017. doi: 10.48550/arXiv.1612.00837.

[7]Y. Qiao, C. Deng, and Q. Wu, 'Referring Expression Comprehension: A Survey of Methods and Datasets', IEEE Trans. Multimed., vol. 23, pp. 4426–4440, 2021, doi: 10.1109/TMM.2020.3042066.

[8]L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, 'Modeling Context in Referring Expressions', ArXiv160800272 Cs, Aug. 2016, Accessed: May 02, 2022. [Online]. Available: http://arxiv.org/abs/1608.00272

[9]L. Yu, H. Tan, M. Bansal, and T. L. Berg, 'A joint speaker-listener-reinforcer model for referring expressions', in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7282–7290.

[10]R. Hu, M. Rohrbach, J. Andreas, T. Darrell, and K. Saenko, 'Modeling relationships in referential expressions with compositional modular networks', in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1115–1124.

[11]L. Yu et al., 'MAttNet: Modular Attention Network for Referring Expression Comprehension', ArXiv180108186 Cs, Mar. 2018, Accessed: Oct. 12, 2021. [Online]. Available: http://arxiv.org/abs/1801.08186

[12]P. Wang, Q. Wu, J. Cao, C. Shen, L. Gao, and A. van den Hengel, 'Neighbourhood Watch: Referring Expression Comprehension via Language-Guided Graph Attention Networks', 2019, pp. 1960–1968. Accessed: Mar. 29, 2022. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Wang_Neighbourhood_Watch_Referring_Expression_Comprehension_via_Language-Guided_Graph_Attention_Networks_CVPR_2019_paper.html

[13]X. Liu, Z. Wang, J. Shao, X. Wang, and H. Li, 'Improving referring expression grounding with cross-modal attention-guided erasing', in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1950–1959.

[14]D. Liu, H. Zhang, F. Wu, and Z.-J. Zha, 'Learning to assemble neural module tree networks for visual grounding', in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 4673–4682.

[15]J. Lu, D. Batra, D. Parikh, and S. Lee, 'Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks', Adv. Neural Inf. Process. Syst., vol. 32, 2019.

[16]W. Su et al., 'Vl-bert: Pre-training of generic visual-linguistic representations', ArXiv Prepr. ArXiv190808530, 2019.

[17]D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, 'IQA: Visual Question Answering in Interactive Environments', 2018, pp. 4089–4098. Accessed: Mar. 23, 2022. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Gordon_IQA_Visual_Question_CVPR_2018_paper.html

[18]P. Wang, Q. Wu, C. Shen, A. Dick, and A. Van Den Hengel, 'Fvqa: Fact-based visual question answering', IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 10, pp. 2413–2427, 2017.

[19]P. Wang, Q. Wu, C. Shen, A. van den Hengel, and A. Dick, 'Explicit knowledge-based reasoning for visual question answering', ArXiv Prepr. ArXiv151102570, 2015.

[20]Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel, 'Visual question answering: A survey of methods and datasets', Comput. Vis. Image Underst., vol. 163, pp. 21–40, Oct. 2017, doi: 10.1016/j.cviu.2017.05.001.

[21]R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, 'From recognition to cognition: Visual commonsense reasoning', in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 6720–6731.

[22]A. Vaswani et al., ‘Attention Is All You Need’, ArXiv170603762 Cs, Dec. 2017, Accessed: Aug. 25, 2021. [Online]. Available: http://arxiv.org/abs/1706.03762

[23]J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, ArXiv181004805 Cs, May 2019, Accessed: Nov. 24, 2021. [Online]. Available: http://arxiv.org/abs/1810.04805

[24]A. Dosovitskiy et al., ‘An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale’, ArXiv201011929 Cs, Jun. 2021, Accessed: Aug. 25, 2021. [Online]. Available: http://arxiv.org/abs/2010.11929

[25]R. Krishna et al., ‘Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations’, Int. J. Comput. Vis., vol. 123, no. 1, pp. 32–73, May 2017, doi: 10.1007/s11263-016-0981-7.

[26]P. Anderson et al., ‘Bottom-up and top-down attention for image captioning and visual question answering’, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6077–6086.

[27]L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, ‘VisualBERT: A Simple and Performant Baseline for Vision and Language’, ArXiv190803557 Cs, Aug. 2019, Accessed: Apr. 29, 2022. [Online]. Available: http://arxiv.org/abs/1908.03557

[28]H. Tan and M. Bansal, ‘Lxmert: Learning cross-modality encoder representations from transformers’, ArXiv Prepr. ArXiv190807490, 2019.

[29]李晓薇.vue.js 前端应用技术分析[J].网络安全技术与应用,2022,(04):44-45.

[30]王伊,王韶红,刘晋泽,吕佳宪.Vue.js 与 Django 组合框架的网络社交系统单页面架构方案设计[J].信息技术与信息化,2020,(01):121-123.

[31]谢思雅,施一萍,胡佳玲,陈藩,刘瑾.基于 Django 的文本情感分类系统设计与实现[J].传感器与微系统,2021,40(11):97-99.

致谢

大学四年以来，感谢各位谆谆教导的老师，使我从中学得各项专业技能，并习得一定程度的专业学习能力，从而对毕业设计各项技术无论熟悉与否都能在较短时间内逐步掌握并把工作做好，其中非常感激田侦老师在我毕业设计阶段悉心教导，对每个阶段的工作存在的问题耐心指导并提出合理建议，对最终论文定稿做了很大帮助，也非常感谢郑州大学以及计算机与人工智能学院平台给我们带来的无限机会，使得我在各项竞赛和创新项目中逐步培养了一些创新和科研意识，提高了遇到问题解决难题的动手能力。最后，再次由衷感谢在我大学四年来对我学业有所帮助的老师和同学们。